Software Specifications

Operating System (COTS) , Redstone DP3R Requirements and Design Specifications

Checkout and Launch Control System (CLCS)

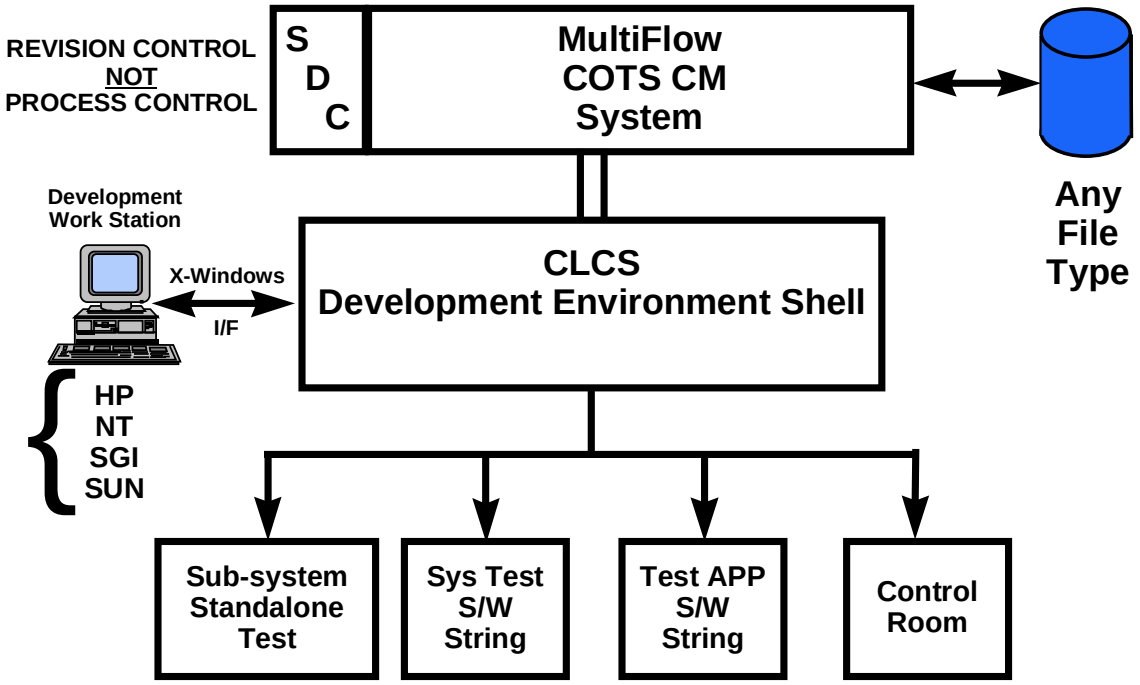
84K00510-210

1. CSC Name

1.1 CSC Name Introduction

1.1.1 CSC Name Overview

In this section provide a short description of the purpose of the CSC, where it is resident, and what it does. The section is in paragraph format. It also contains a Conception Level data flow diagram. The words in the paragraph should refer to the blocks in the diagram by way of explanation. The following diagram is a conception level diagram.



1.1.2 CSC Name Operational Description

In this section provide a short description of how the CSC does what it does. This section can be thought of as a tutorial on how the CSC works. It may make reference to the data flow diagram and should address all of the interfaces and, at a high level, the data that flows across each.

This section is in paragraph format.

1.2 CSC Name Specifications

1.2.1 CSC Name Groundrules

Provide a bulleted list of groundrules and assumptions that relate to the CSC. Things that should be included in this section include:

- Things that the CSC will not do that someone might expect it to do.
- Things that you expect to be provided to you by:
  - Another CLCS CSCI or CSC.
  - External system H/W or S/W interface (identified by ICD number).
  - New CLCS H/W interface (identified by specification).

- Any limitations or constraints that are placed on the CSC.

  Note: Software interface requirements identified in the groundrules for one CSCI/CSC usually impose requirements on other CSCI/CSCs which must be documented in the functional requirements of those CSCI/CSCs

First, second, third, and fourth level bullets should follow the following convention:

- First Level Bullet #1. Bullet - 12 point.
- First Level Bullet #2. . Bullet - 12 point.
  - Second Level Bullet #1.
  - Second Level Bullet #2.
    - Third Level Bullet #1. Bullet - 8 point.
    - Third Level Bullet #2. Bullet - 8 point.
      - Fourth Level Bullet #1. Bullet - 8 point diamond.
      - Fourth Level Bullet #2. Bullet - 8 point diamond.
  - Second Level Bullet #3.
- First Level Bullet #3. . Bullet - 12 point.

1.2.2 CSC Name Functional Requirements

Provide a numbered list of functional requirements, organized by major/minor function, that describe how this CSC performs its intended functionality. These requirements must be written in concise, unambiguous language; each requirement must be stated as a complete sentence. Requirements must be stated in quantifiable, measurable parameters, or actions to be taken when a set of conditions is satisfied. The section should be organized in a logical manner (e.g., by major & minor function) rather than by placing requirements in random order within it. Things that should be included in this section include:

- Major functions of the CSC.
- What each function does. This is a list of requirements that state what the function does.
- All Inputs the CSC responds to and what it does with each.
- Also describe, where applicable, each of the following
  - Errors that the CSC will trap.
  - The CSC's reaction to each of the errors.
- Any data that the CI uses that is adaptable (e.g., user profiles or user selected options, coefficients used to calculate linearization from table build or the OLDB).
- Critical processing requirements (e.g., if a CSC has priority as well as normal queuing to perform some processing at a higher than normal priority these must be specified).
- If the CSC has multiple operating modes the modes are described and a matrix included which identifies which functions or requests are legal in each mode.

Follow the numbering convention described below:

Example

The Functional Requirements for CSC Name are arranged in the following major/minor functions:

1. File Repository Management
2. Software Download
3. Platform Configuration and Initialization
4. Activity Management

5. Node Configuration
6. Recording
7. Data Requirements

1 File Repository Management

1.1 OPS CM shall provide the following file repositories:
1.1.1 Certified Application Repository (CAR)
1.1.2 Uncertified Application Repository (UAR)

1.2 OPS CM shall allow for the preservation of the following file attributes for any files manipulated by an OPS CM process (as supported by the OS platform):
a. File name
b. Individual owner

1.3 OPS CM shall allow for the preservation of the following directory attributes for any directories manipulated by an OPS CM process (as supported by the OS platform):
a. Directory name
b. Individual owner

1.4 OPS CM shall provide the capability to introduce a temporary baseline into an OPS CM repository.

2 Software Download

2.1 OPS CM shall provide a capability for a CLCS platform to be loaded with an SCID baseline.

2.2 OPS CM shall provide a capability for a CLCS platform to be loaded with an TCID baseline.

2.3 OPS CM shall provide a capability for an HCI platform to be loaded with user and positional home directories.

2.4 OPS CM shall provide an HCI GUI to allow authorized users to request and execute download functions.

1.2.3 CSC Name Performance Requirements

Provide a bulleted list of performance requirements that describe the performance requirements for the CSC. These can be described in terms of the threads that are involved and their required performance. The performance requirements must be stated in quantifiable, measurable parameters. These requirements should be stated in the same order as major/minor functions above and should use the same convention for major/minor function headings.

1 Node Configuration

1.1 OPS CM shall be able to load and initialize a new TCID on a test set of up to ***TBD*** nodes within 15 minutes.

Note: Initialization time ends when the OPS CM process initiates the user application software start-up script(s) for HCI, CCP, and DDP. It ends for Gateways when the Gateway moves into the operational mode (time constraints assume there is no man-in-the-loop for the combined download & initialization process.

1.2 OPS CM shall load SCID and TCID software on a launch configuration of up to ***TBD*** nodes in 2 hours or less.

1.2.4

CSC Name Interfaces Data Flow Diagrams

This section provides a description and diagram of all of the interfaces to the CSC. Describe in words and lists the interfaces. Use diagrams like the following to depict the interfaces to the CSC.

External Data Flow Diagram Example

The purpose of the External Interface diagram is to show the interfaces between the CSC and the external elements it interfaces with. It conveys in a pictorial format all of the input and output streams that the CSC deals with, but not their content. Include with the diagram a short paragraph describing the data flow so that a reader can pick up the document and understand the data coming into the CSC and the data sent out of the CSC without a conversation with the developer.

Note: Do not use the underscore_character between words. This is an English language document that we want people to be able to pick up and read in their natural language, not a pseudo programming language.

Note: This is the end of the Design Panel 2 Required material. The information covered below is for Design Panel 3.

1.3

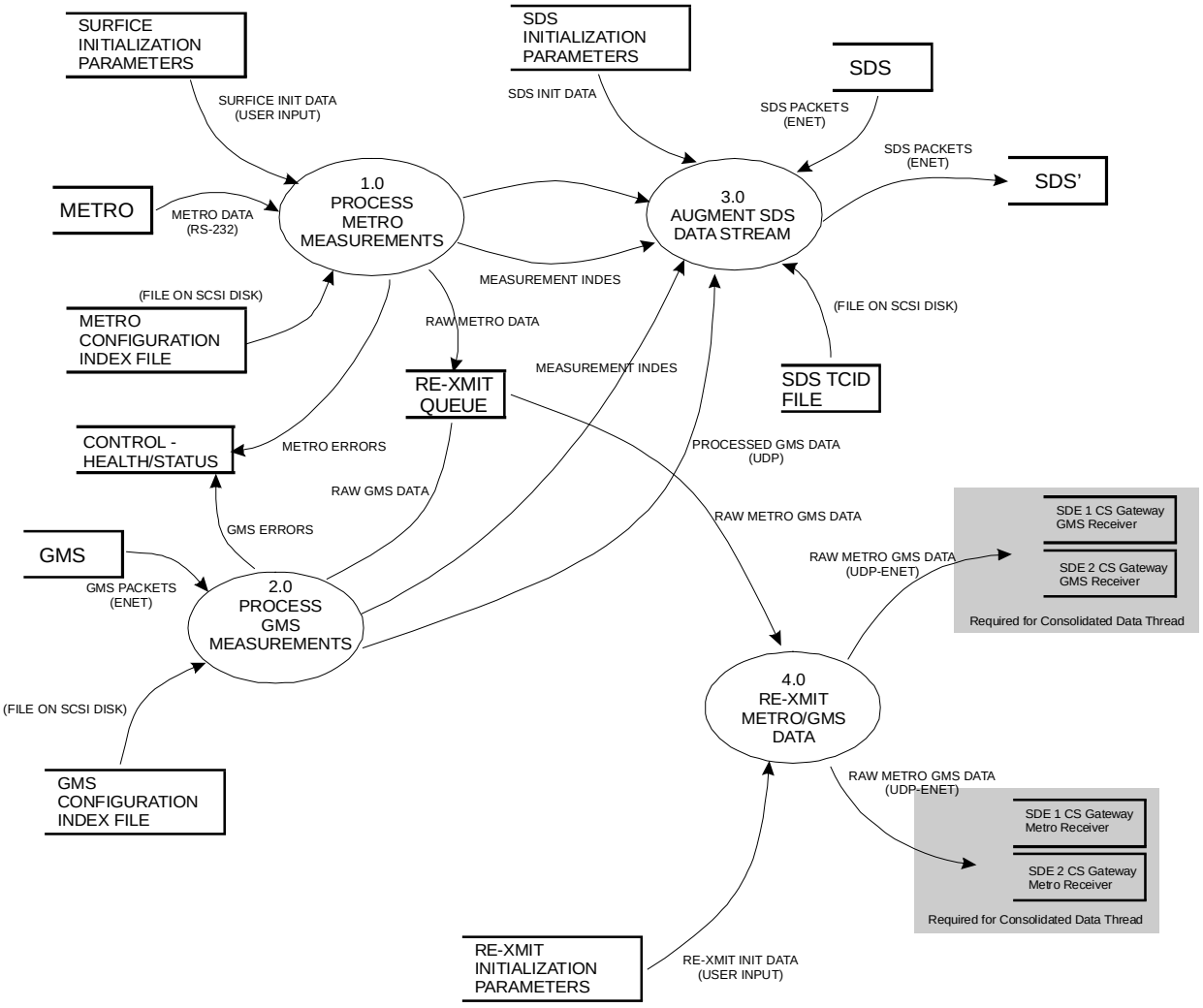
CSC Name Design Specification

Include a brief description of the architecture of the CSC. If there are any priorities associated with the design they should be specified here.

1.3.1 CSC Name Detailed Data Flow

This data flow provides a pictorial representation of the data flow between external sources and destinations and the major and minor functions of the CSC. This is an example detailed data flow diagram.

Detailed Data Flow Diagram Example



The purpose of the Detailed Interface/Data Flow diagram is to show all of the interfaces, internal as well as external, of the CSC. It conveys in a pictorial format all of the input and output streams that the CSC deals with, but not their content. The content of each of these streams of data is described in the detailed design below. Include with the diagram a short paragraph describing the data flow so that a reader can pick up the document and understand the data flow without a conversation with the developer.

Note: Do not use the underscore_character between words. This is an English language document that we want people to be able to pick up and read in their natural language, not a pseudo programming language.

1.3.2 CSC Name External Interfaces

1.3.2.1 CSC Name Message Formats

This data is the System Messages output by the CSC.

Example:

Message Number = _____
Message Group = _____

GSE Gateway **I1** Adapter Failure - Polling Stopped - Switchover Requested
Status Register Contains H**#I2#** S/B H**#I3#**

Insert #1 = One Character Integer Value 1 through 8
Insert #2 = Four Character Hex Value
Insert #3 = Four Character Hex Value

Help Information Content:

Help information content will contain a breakdown of the status register showing the meaning of each bit. This file would also show (in time) the likely cause of different failure indications.

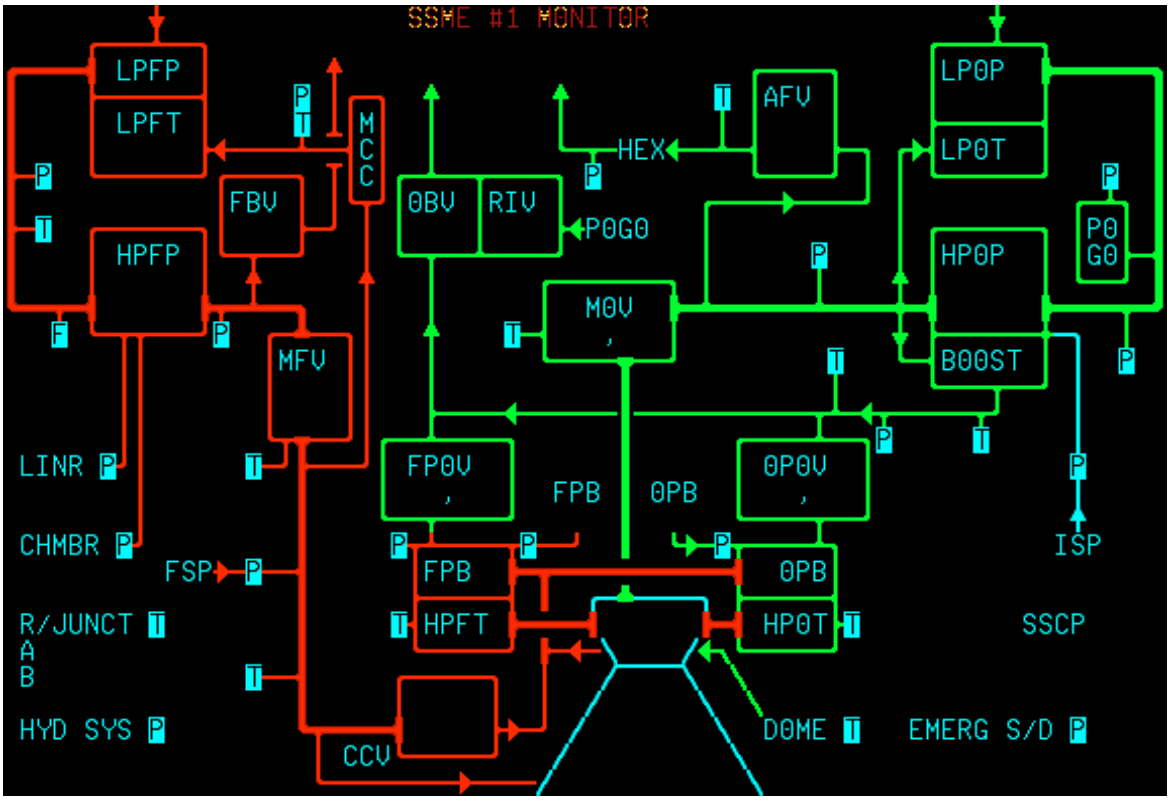
Details Information:

Details Information will be available on the Status Register only.

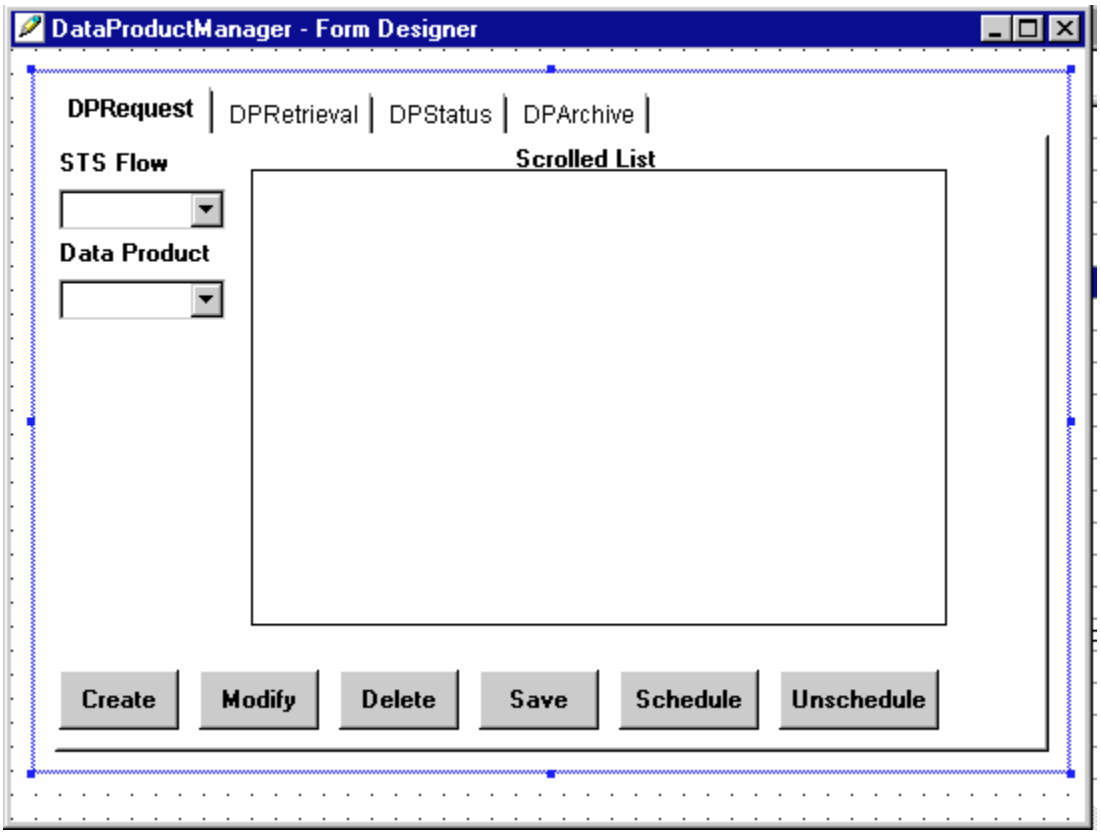
1.3.2.2 CSC Name Display Formats

This is the design of Displays produced by the CSC.

Example 1:



Example 2:



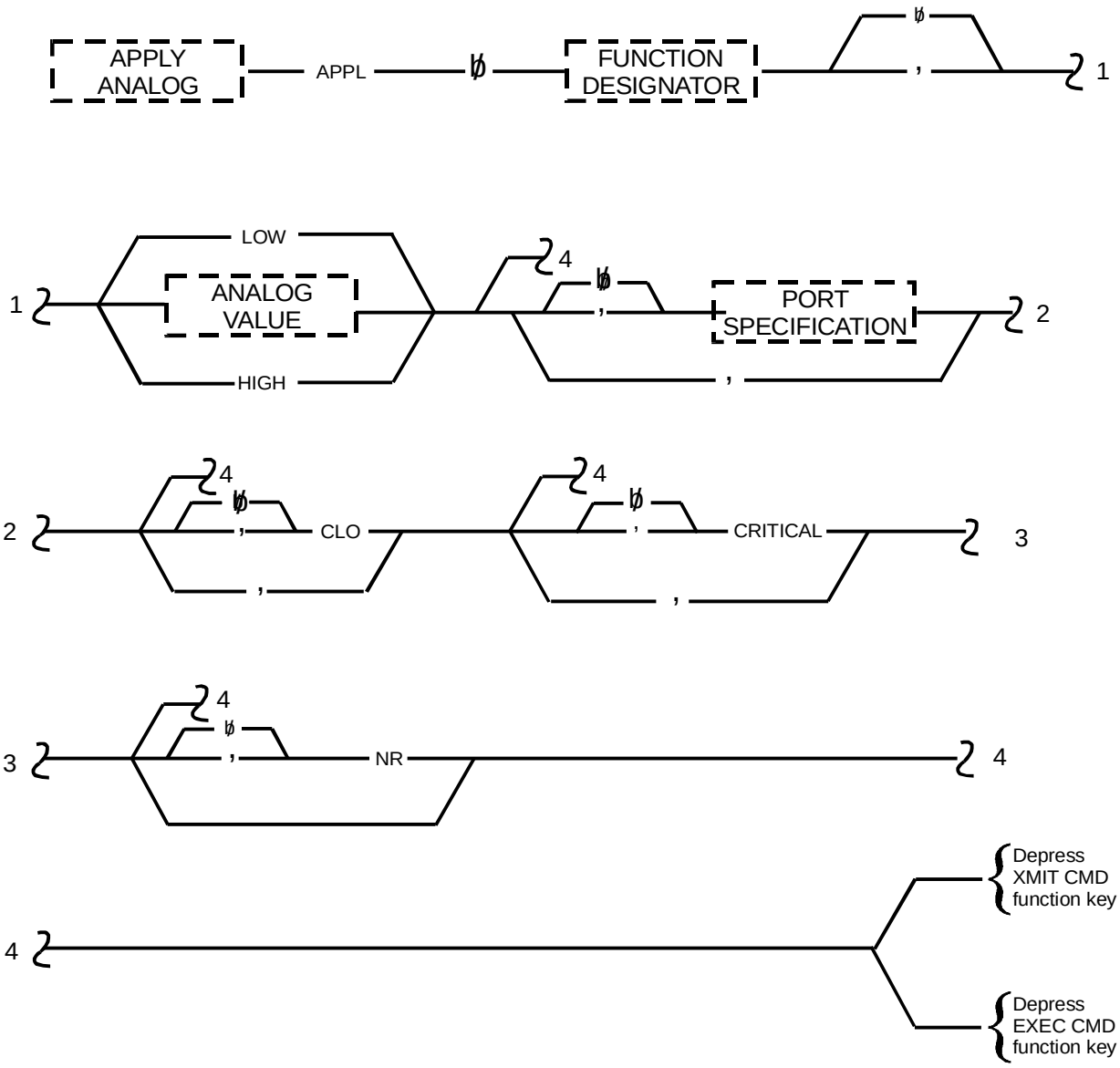
1.3.2.3

CSC Input Formats

This data is for CSCs that have a language like interface (e.g., Command Processor — Apply Analog)

Example:

APPLY ANALOG



1.3.2.4 CSC Name Printer Formats

This data documents the design of anything printed on printers.

Example:

LDB MASS MEMORY READ

```
CDT=-99:2359/59  GMT=132:1232/24.375  GATEWAY=LDBA  CMD=MASS MEMORY READ  FCNL DEST=MM  TRNS ID=01E7  CPU=SDC
        FTSB=  5  1  4  4          DEST=MM 1     SEQ     BLKS=9

MASS MEMORY RESPONSE  TIME=1232/24.531 GMT FUNCTIONAL DESTINATION= MM      TRANS ID= 01E7          CAPABILITY 1 READ OF
        MM2    FILE 0   TRACK 6          SUBFILE          7          BLOCK 0          BLOCK# 30 0F 31 BLOCKS          READ ERRORS

0000    C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6   C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6   000F
0010    C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6   C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6   001F
0020    C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6   C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6   002F
  •
  •
01F0    C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6   C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6   01FF
```

LDB MASS MEMORY WRITE

```
CDT=-99:2359/59  GMT=132:1245/02.465  GATEWAY=LDBA  CMD=MASS MEMORY WRITE  FCNL DEST=MM  TRNS ID=02F0  CPU=DPS
        FTSB=  5  1  4  4          DEST=MM 1     SEQ     BLKS=3          DATA:
0000    0000

MASS MEMORY RESPONSE  TIME=1245/02.900  GMT          FUNCTIONAL DESTINATION = MM      TRANS ID=02F0
          CAPABILITY 1 WRITE TO MM1          UNSUCCESSFUL
```

LDB MMU READ LOAD BLOCK

```
CDT=-99:2359/59  GMT=132:1247/24.449  GATEWAY=LDBA  CMD=MMU READ LD BLCK  FCNL DEST=MM  TRNS ID=02A7  CPU=SDC
        PH/LD BLK = #0302
MM VERSB=#0000  PATCH ID=#0000   FSW ID=  8       CHKSM #43A9  MM1    NOCOMP          DUMP

MASS MEMORY RESPONSE  TIME=1247/12.950  GMT          FUNCTIONAL DESTINATION = MM      TRANS ID=02A7
        CAPABILITY 2 READ & DUMP LOAD BLOCK  MM1       PHASE=01        LOAD BLOC0A             W/O COMPARE
        STATUS= SUCCESSFUL
        FILE  3  TRACK  6        SUBFILE  1       BLOCK  9        BLOCK COUNT= 31
0000    9E31  B5F6  0001  C6C6  C6C6  C6C6  C6C6  C6C6   C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6   000F
  •
01F0    A1F3  3089  A1F3 C6C6  C6C6  C6C6  C6C6  C6C6   C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6  C6C6   01FF
```

1.3.2.5 Interprocess Communications (C-to-C Communications?)

This is the data that is sent between CSCs that may or may not be in the same processor. Examples are the C-to-C formats that Tom Jamieson has developed for us. These formats are prepared initially by the Systems Engineering Team (Tom Jamieson) but not owned by the SEI group during the design of the CSCs that send and receive them. The SW design team is expected to

1) Assume control of the design of the C-to-C
2) Develop the design
3) Document the design — in this document?
4) Provide the design to Tom Jamieson after DP3 approval for incorporation into the System Interface Document (TJJ is this name correct?)

Example:

Change Data Packet Payload Body Entry (Analog)

B15 B14 B13 B12	B11 B10 B9 B8 B7 B6 B5 B4 B3 B2	B1 B0
0000-1001 *	Status	FDID MSB
FDID - 16 LSB		
CC		
EU**		
EU**		

* = "this is an analog entry and 0000-1001 = 100 usec time".
** = 32 bits

1.3.2.6 CSC External Interface Calls (e.g., API Calling Formats)

This is the data that is sent between CSCs via a calling mechanism (e.g., API call)

Example:

TBS

1.3.2.7 CSC Name Table Formats

This data documents the design of the tables used internal to the CSC and provided from an outside source (e.g., Gateway Table Build, OLDB, etc.)

Example:

TBS

1.3.3 CSC Name Test Plan

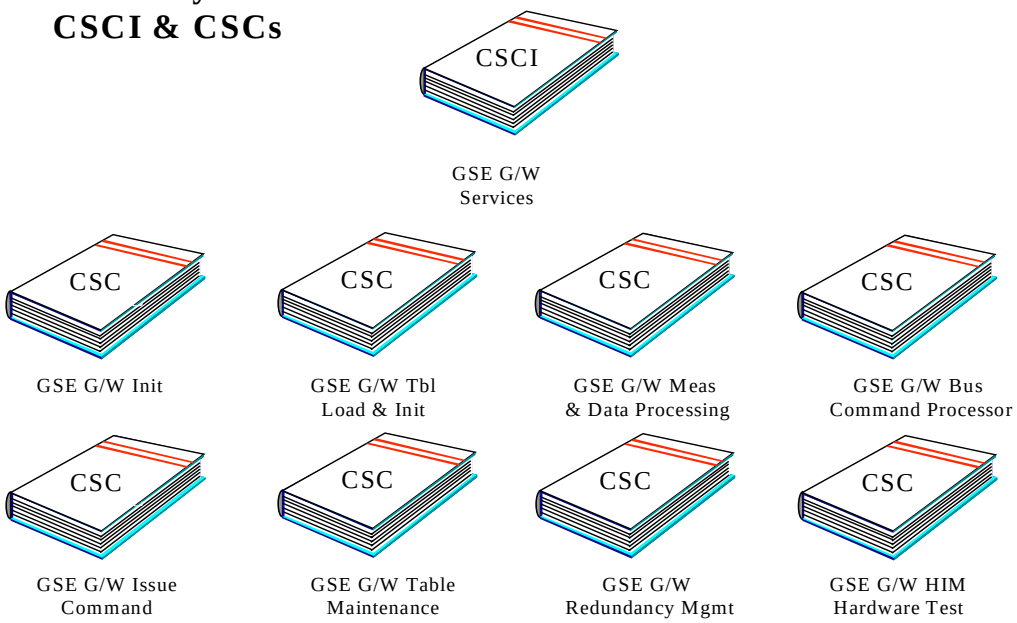
Example:

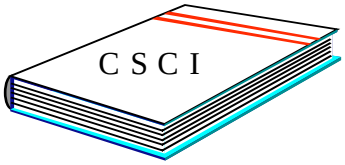
TBS

Appendix A

In most cases a CSCI is composed of the functionality contained in multiple CSCs that are part of the CSCI. The intent of the SRS/Design Specification is to capture the requirements and design of both the CSCI and its CSCs in one set of volumes. Since we are using electronic forms of documentation for the most part these should be easily captured in one place with links to others? At any rate one option for the structure of a CSCI document is as follows:

GSE Gateway Services
CSCI & CSCs





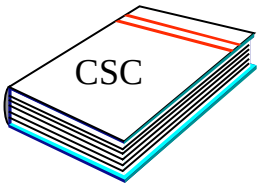
GSE G/W Services

CSCI Outline

1.0 CSCI Name
1.1 CSCI Name Introduction
1.2 CSCI Name Overview
1.2.1 CSC 1 Name Document
1.2.2 CSC 2 Name Document
1.2.3 CSC 3 Name Document
o
o
o
1.2.N CSC N Name Document

Design Panel 2

CSC Requirements Spec

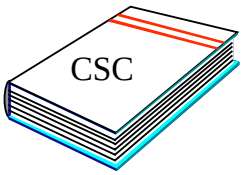


GSE G/W Init

1.0 GSE G/W Init
1.1 GSE G/W Init Introduction
1.1.1 GSE G/W Init Overview
1.1.2 GSE G/W Init Operational Description
1.2 GSE G/W Init Specifications
1.2.1 GSE G/W Init Groundrules
1.2.2 GSE G/W Init Requirements
1.2.3 GSE G/W Init Performance Requirements
1.2.4 GSE G/W Init Interfaces/Data Flow Diagrams

Design Panel 3

CSC Requirements Spec



GSE G/W Init

1.0 GSE G/W Init
1.1 GSE G/W Init Introduction
1.1.1 GSE G/W Init Overview
1.1.2 GSE G/W Init Operational Description
1.2 GSE G/W Init Specifications
1.2.1 GSE G/W Init Groundrules
1.2.2 GSE G/W Init Requirements
1.2.3 GSE G/W Init Performance Requirements
1.2.4 GSE G/W Init Interfaces/Data Flow Diagrams

Present only Changes

CSC Design Spec

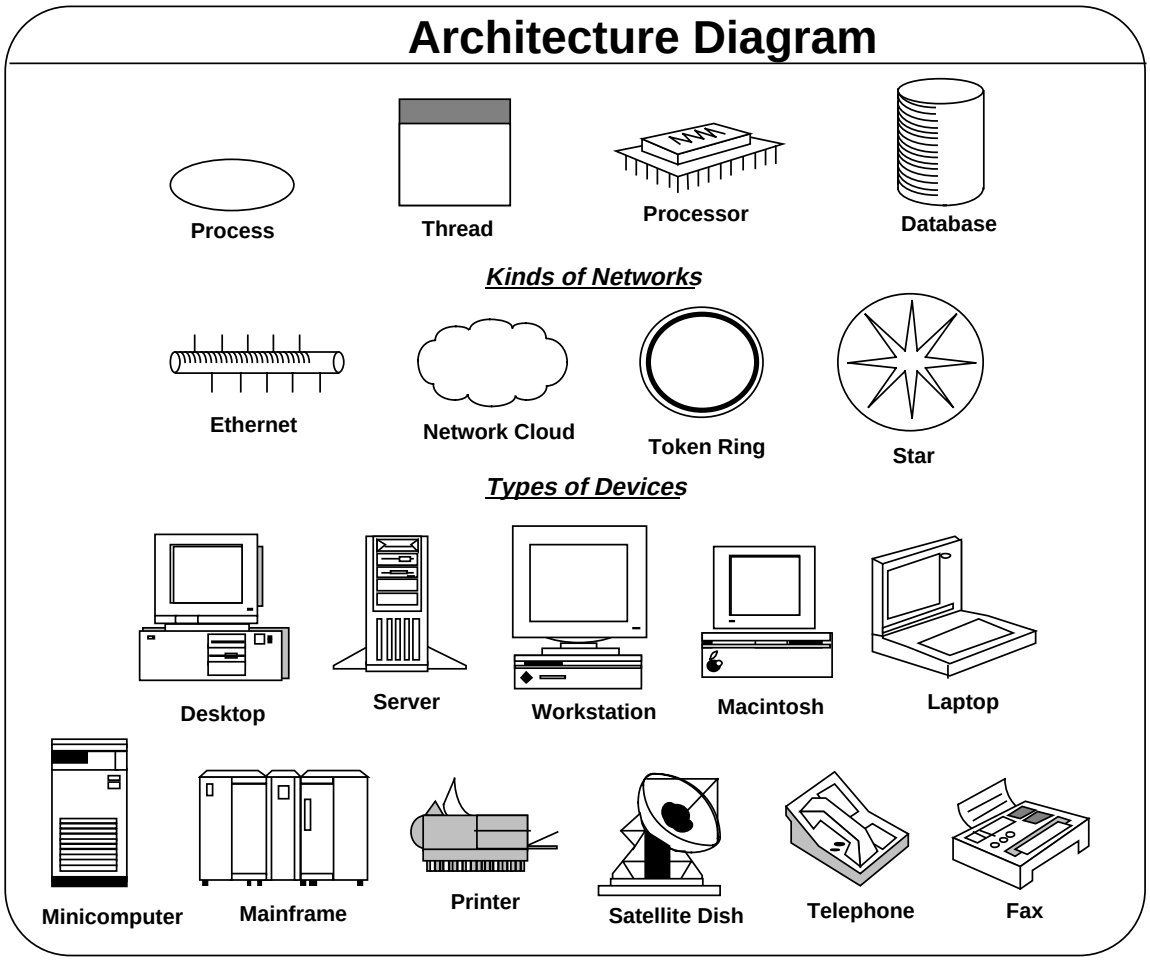
1.3 GSE G/W Init Design Specification
1.3.1 GSE G/W Init Detailed Data Flow
1.3.2 GSE G/W External Interfaces
1.3.3 GSE G/W Internal Interfaces
1.3.4 GSE G/W Structure Diagram
1.3.5 GSE G/W Test Plan

Appendix B

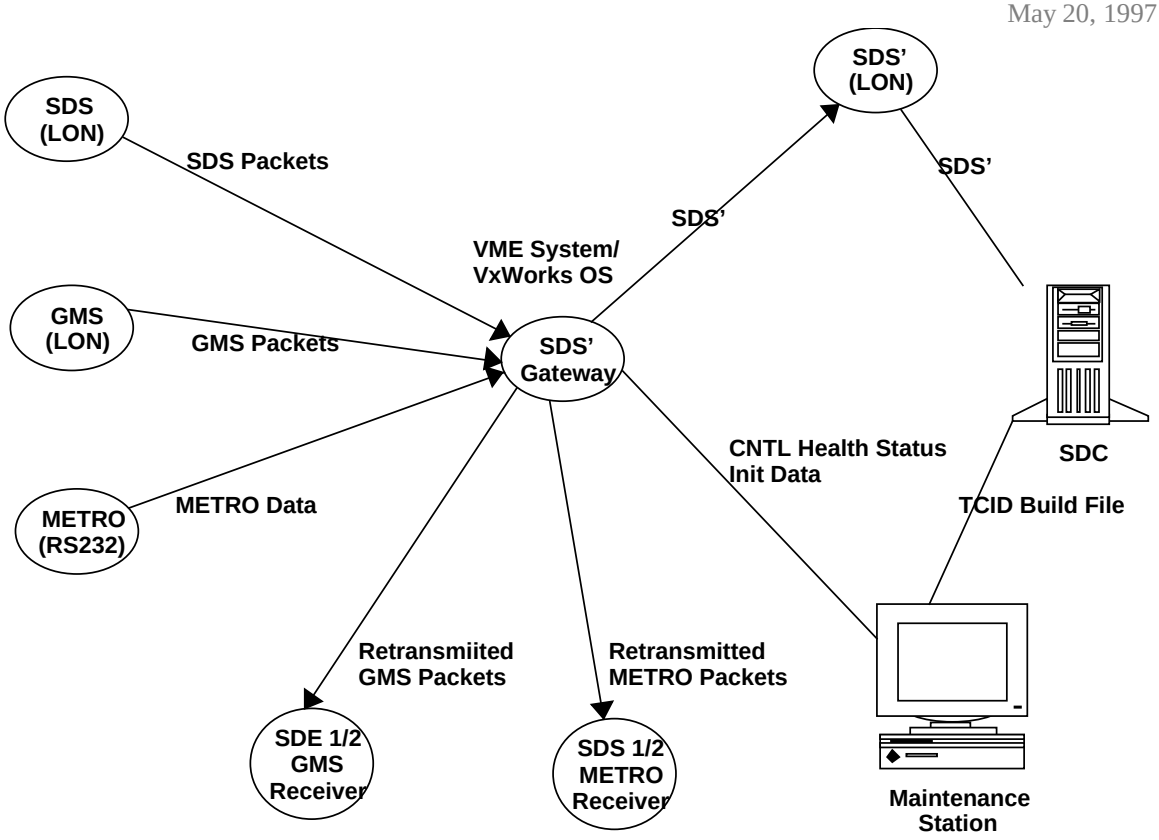
This appendix provides information that will be used for CSCs that are developed using Object Oriented Design (OOD) methodology.

Note: The information in Appendix B is preliminary and will change as the OOD methodology to be used on the CLCS project matures.

The OMT/Rumbaugh notation will be used to document the architecture according to the following legend:



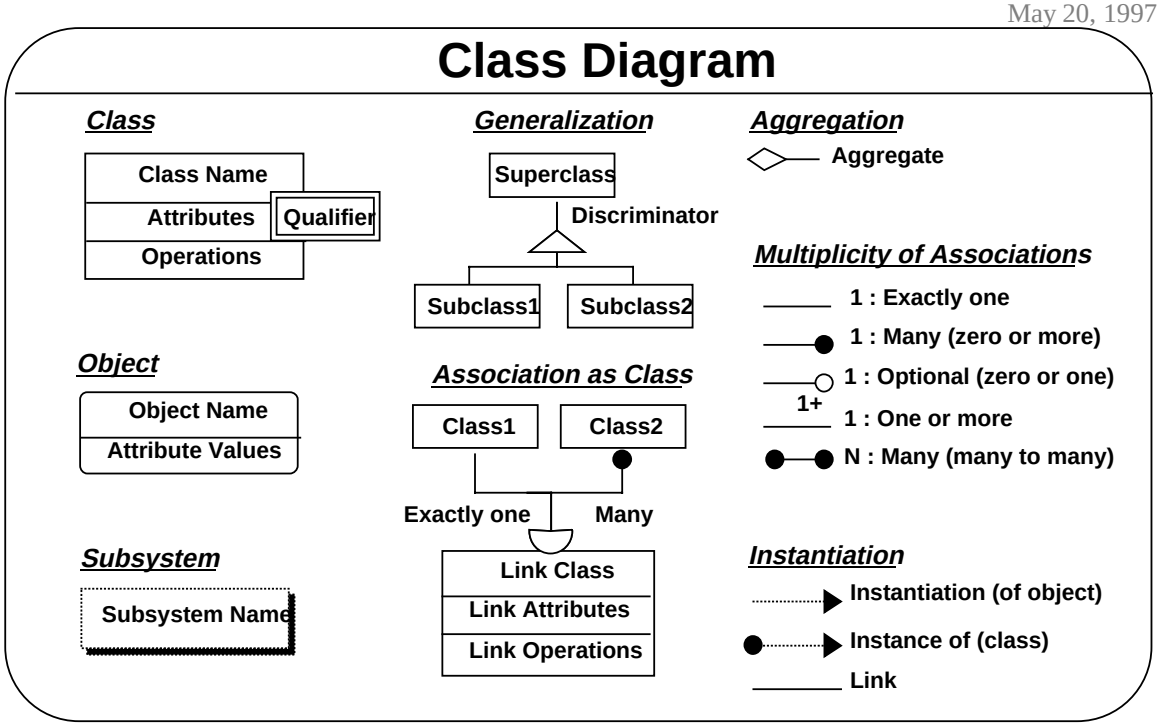
An example architecture diagram is shown below:



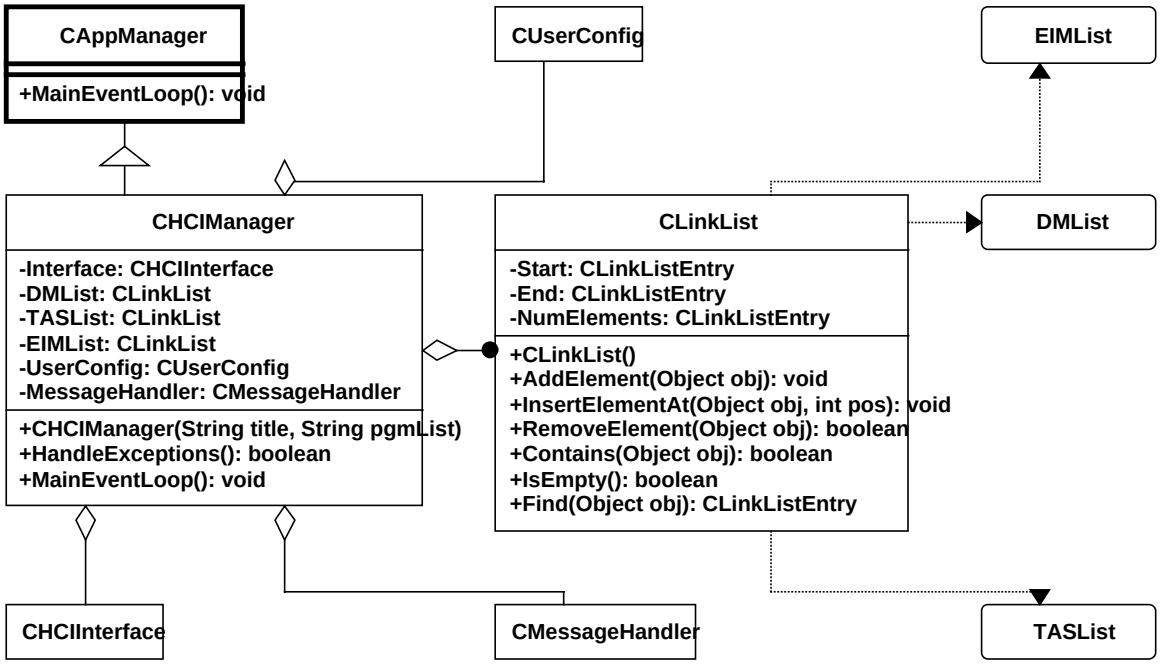
1.4 DP2

Prior to DP2, the designer will decide whether to use Structured Design or OOD. If structured design is to be used, the examples used in the body of this document will be used. If OOD is to be used, the CSC designer should provide a set of preliminary class diagrams. At DP2, the focus is on requirements and preliminary design. Accordingly, the class diagrams presented at DP2 should describe the key abstractions in the requirements ("problem space" classes).

The OMT/Rumbaugh notation will be used to document class information according to the following legend:



An example class diagram is shown below:

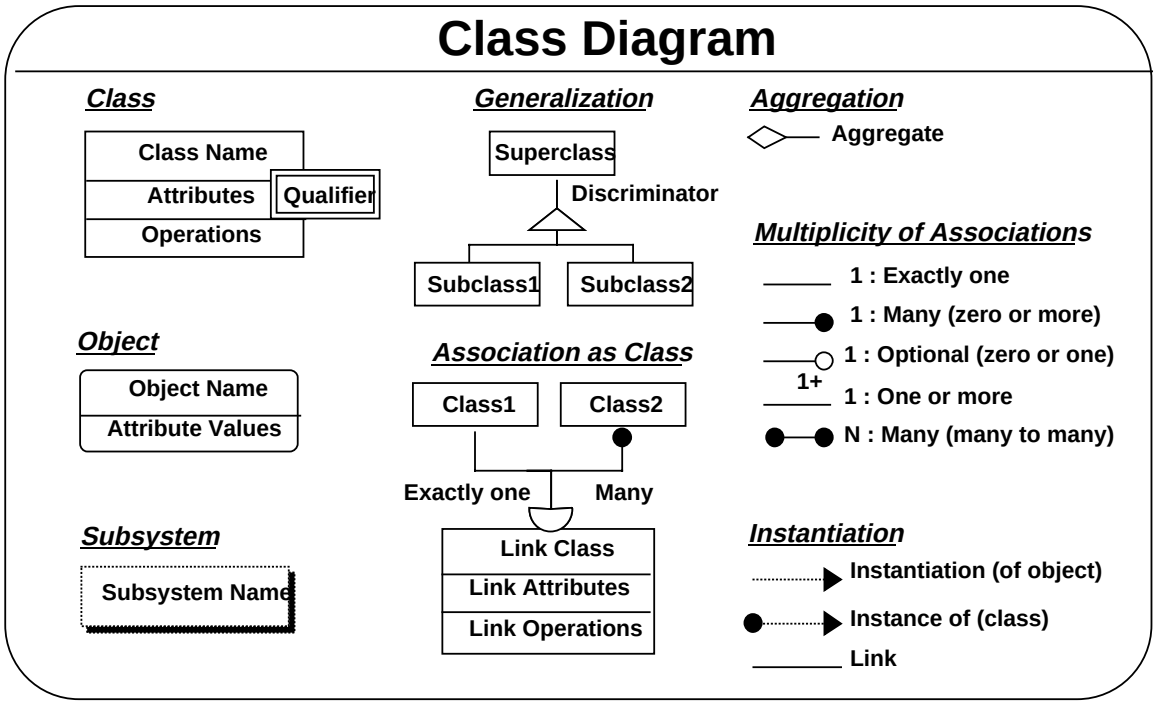


1.5 DP3

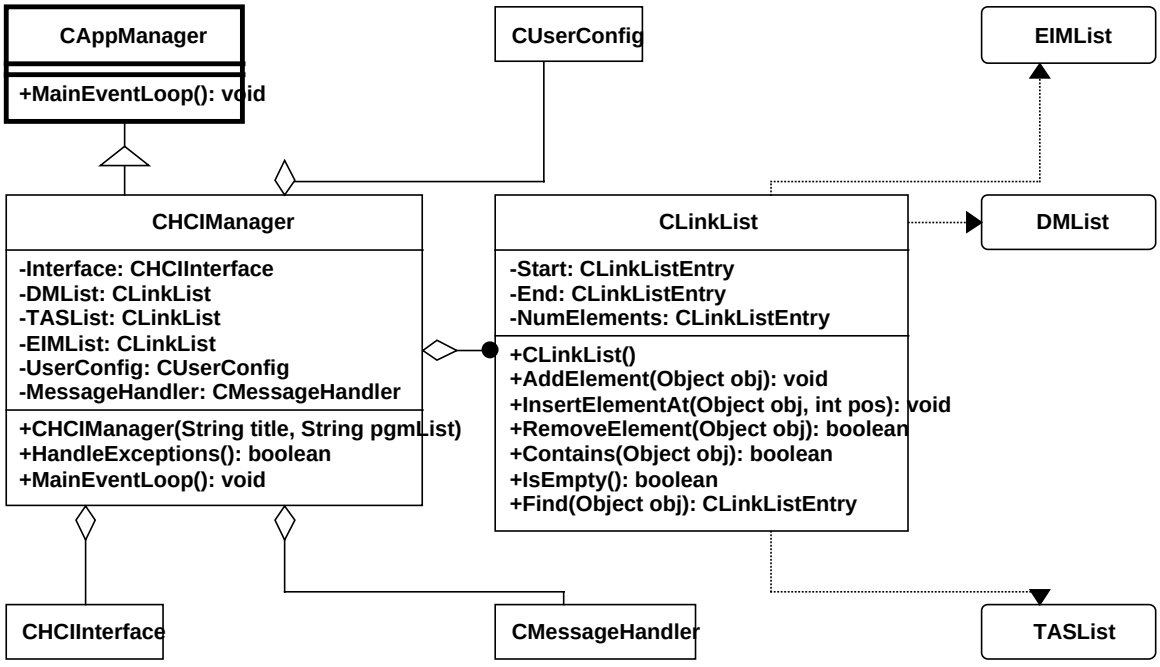
For DP3, the designer should provide a set of additional class diagrams that augment the analysis phase classes presented at DP2. Further refinements of the classes presented at DP2 may be included to provide context and aid

in understanding. The purpose of these diagrams are to document the mechanisms that will be used to implement the requirements for this CSC.

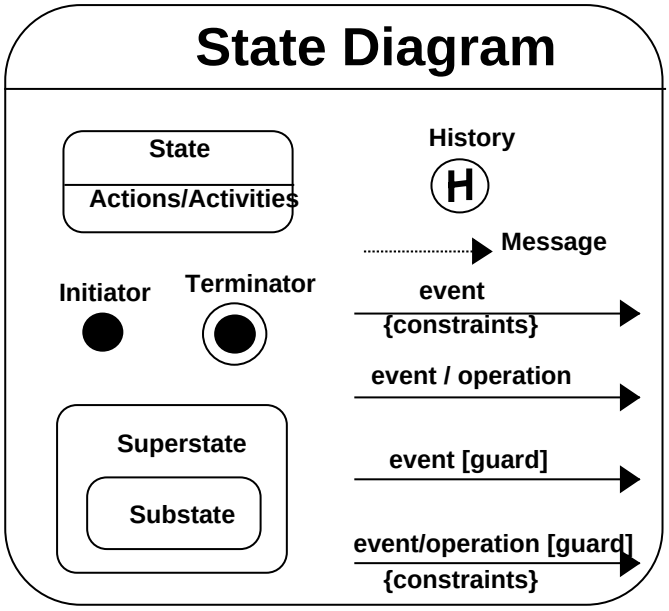
The OMT/Rumbaugh notation will be used to document class information according to the following legend:



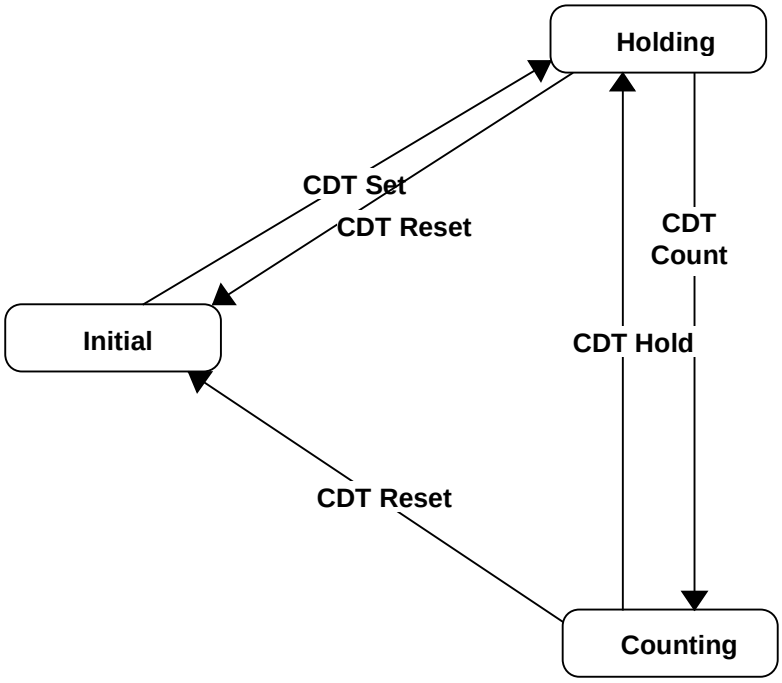
An example class diagram is shown below:



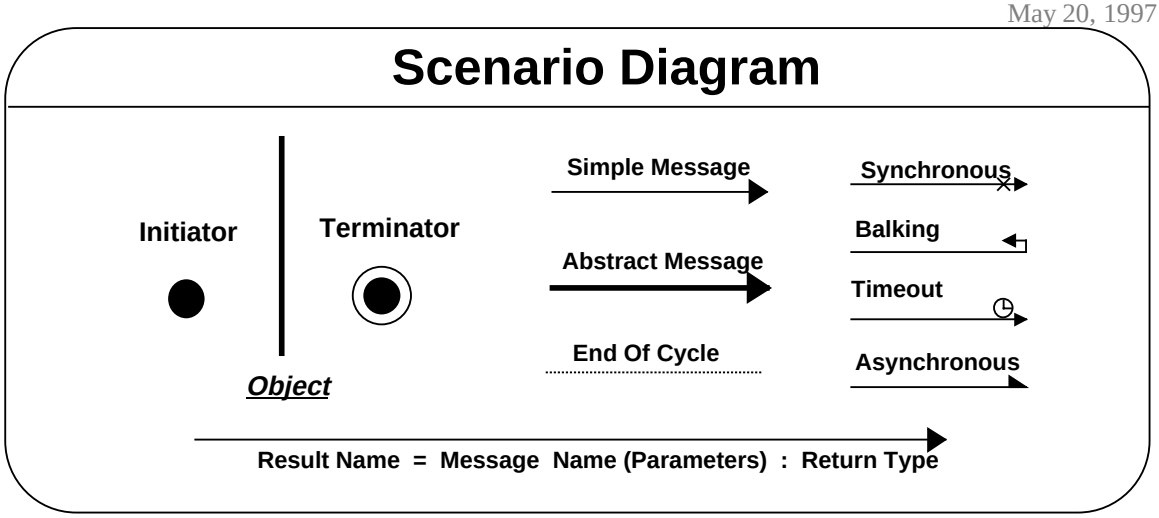
In addition, for classes that exhibit state machine behavior, a state diagram should be included that depicts the dynamic behavior of the class according to the following legend:



An example state diagram for a software implementation of a countdown clock is shown below:



In addition, key scenarios of the CSC's operation or key processing threads should be documented with an scenario diagram according to the legend below:



An example scenario diagram that illustrates the message flow responsible for opening a specific valve is shown below: